UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/673,587	09/29/2003	Jonathan Appavoo	YOR920030317US1 (16856)	1607

23389 7590 09/09/2008

SCULLY SCOTT MURPHY & PRESSER, PC
400 GARDEN CITY PLAZA
SUITE 300
GARDEN CITY, NY 11530

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
09/09/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/673,587	APPAVOO ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 July 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 7/01/08.

As indicated in Applicant's response, claims 1, 5, 7, 10, 13, 17, 18, 23 have been amended. Claims 1-24 are pending in the office action.

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).
A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 10, 18 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 7 of copending Application No. 11/227,761 (hereinafter '761).

Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following observations. Following are but a few examples as to how the certain claims from the instant invention and from the above copending application are conflicting with each other.

As per instant claims 1 and 18, '761 claim 7 also recites dynamically update an operating system via hot swapping a new object for its old instance object; and although '761

claim 7 does not recite replacing while the operating system remains active to provide continual availability of resources to applications in the computer system, this 'hot swapping' is deemed obvious variant language for dynamic operation of an OS *while said operating system remains active ... provides continual availability of hardware resources by applications in the computer system.* Moreover, '761 claim 7 does not explicitly recite *instantiating* instance of new code to replace first code, and establishing a *quiescent state of said first component*, and transferring state of first component to the new code. But in view of '761 claim 1 reciting loading of a new object with identifying of *version of old object* with said *new object*, the instance of the update or new code component to replace the old object suggests instantiating a object and loading it, and this is rendering the instantiating step an obvious variant of loading instance of version of new object as recited above. As for instant claim 1 *replacing* step (replacing identified references to first code component with those of new component) '761 recites 'changing reference pointer' from the old object to the new object; that is, this reference pointer would be equivalent to *identifying references* to said first code component and *replacing the identified references to said first code component to said new code component.* As for the quiescent state limitation, '761 does recite determining 'when instance of old object reaches a safe point prior to changing the reference pointer'; hence this is deemed an obvious variant of the 'quiescent state' establishing by instant claim 1.

As per instant claim 10, these claims recite the main limitations of instant claim 1, hence would also have been obvious variations of '761 claim 7 in light of the above analysis.

Specifications Objections

4. The specifications for reciting 'source code' component (SUMMARY, pg 3-4) which in light of the Clustered Object at runtime being identified by table pointers appears to have been a misused terminology. There is no *source code* per se being generated in the hot swapping of a running OS as depicted in the entire Disclosure. Absent explicit definition of the 'source code' any where in the Specifications, this syntactic usage is deemed not consistent with accepted meaning of the above term; and this language is objected to or not being commensurate with accepted lexicography. Correction is required.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 10-11, 13-17 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

The current focus of the Patent Office in regard to statutory inventions under 35 U.S.C. § 101 for method claims and claims that recite a judicial exception (software) is that the claimed invention recite a practical application. Practical application can be provided by a physical transformation or a useful, concrete and tangible result. The following link on the World Wide Web is the United States Patent And Trademark Office (USPTO) reference in terms of guidelines on a proper analysis on 35 U.S.C. §101 rejection

<http://www.uspto.gov/web/offices/pac/dapp/opla/preognotice/guidelines101_20051026.pdf>

Specifically, claim 10 recites a system for swapping source code in an OS system, comprising *means for* instantiating, establishing, swapping, identifying, transferring and executing the swapping. The Disclosure teaches using C++ language as means to implement the mechanism for hot swapping based on Clustered Object and using set of tables (see Specifications, pg. 5-6) as well as Mediator Object (Specs, pg. 10). The recited means for identifying and replacing amount to functionality based on software-implemented OO code; thus the means as claimed are devoid of support by any hardware in order to carry out said software functionality. As set forth in the USC 101 Guidelines (Annex IV, pg. 52-54) the claimed means are merely listed 'functional descriptive material' and as such cannot be materialize via hardware-based execution to yield a tangible, useful and concrete result. Besides, software listing per se does not constitute any of the 4 categories of statutory subject matter. The claim is rejected for not being a statutory category and for remaining a non-practical application.

Claims 11, 13-17 are rejected for failing to provide hardware support to embody the above means or to execute the software functionality thereof.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 10-17 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 10 recites "to replace *said first code component*" (line 6), then '*the* first source code component' (lines 8-9) and later, '*a first source code* component' (line 11) whereas only 'one code component' or 'source code' is introduced in the preamble. There is insufficient

antecedent basis for 'said first code component' and 'the first source code component' in the claim; and this is rendering the 'a first source code component' (line 11) even more indefinite because this introduces a lack of structural relationship with the above limitations. One would not be able to see whether 3 components exist distinct from each other or just one instance of first code component. Lacking a clear language showing how the *metes and bounds* requirement of the claimed features sis met, the above limitations will be given weight only to the extent of a code component to be replaced by a newer version thereof. Claim 10 is also rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential structural cooperative relationships of elements, such omission amounting to a gap between the necessary structural connections. See MPEP § 2172.01. The omitted structural cooperative relationships are: how first component code being swapped in the running OS relates to 'the first source code component' and 'a first source code component'. Claim 10 is rejected along with dependent claims 11-17 for lack of antecedent basis and for omission of structural relationship.

9. Further, claims 13-17 are also failing the *metes and bounds* requirement for reciting 'first code component' which does not enable one of ordinary skill in the art to make sense of any relationship between *first code component* and *the first source code* (or *a first source code component*) or *said first code component* set forth above as devoid of proper antecedent basis in the base claim.

10. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

11. Claims 10-17 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, Claim 10 is rejected for not being provided with proper support in reciting 'source code component' in a computer system. This *source code* component is deemed not disclosed as a non-executable human readable format in the Disclosure; that is the terminology termed as 'source code' in the entire disclosure is not deemed realistically represented by its lexicographic and well-accepted meaning, lacking explicit definition for it in the Specifications. The claim recites swapping (first and new) source code component, and one of ordinary skill would not be able to construe hot swapping of OS code components (COID – Fig. 1 – Note: hot swapping and state transfer entail a executing environment of machine level objects, no longer for a source code development; e.g. Hot-swapping Overview, pg. 6-8) as extensively depicted in the Specifications reasonably conveys so that component as 'source code' is actually being replaced or swapped via state transfer as construed from reading the Specifications. The above language amounts to a lack of description in that the inventor is deemed not in possession of the limitation at the time the invention was made. Further, the undefined relationship between the 'source code' component and 'first code' component as recited (refer to USC 112, 2nd paragraph) makes it even more difficult how the claim as a whole can been exactly matched with proper parts of the Disclosure. The 'source code' limitation will be given no patentable weight.

Claims 11-17 fail to remedy to the above lack of proper description deficiency and are also rejected.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 1-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Slingwine et al., USPN: 6,219,690 (hereinafter Slingwine).

As per claim 1, Slingwine discloses in a computer system using an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first code component, (e.g. data coherence … hardware requirement … cache memories -col. 5, lines 4-20; shared-memory - col. 6, lines 41-61), a method of replacing said first code component with a new code component, the method comprising:

instantiating an instance of a new code component (e.g. *generation, generation data structure* - col. 7 lines 18-29 – Note: updating data structure and enabling new element to be created in data structure reads on instantiated new element – see col. 8 lines 31-67 – in table structure of generation component and generation callbacks – see *next generation* - col. 12 lines 35-53) to replace said first code component;

establishing a quiescent state for the first code component (e.g. col. 7, lines 50-67);

transferring state from the first code component to the new code component (e.g. *unlinking ... adds a callback to the next generation... links the new element ... erases the*

original element – col. 8 lines 54-67; *is cleared ... ready for the next generation* – col. 11 lines 10-17); and

swapping the new code component for the first code component, including:

identifying references to said first code component (e.g. current generation 108 – pg. 3 – Note: CALLBACK processing to make next generation to become current generation – see col. 6, lines 41-50 – whereby associated structured context information for a thread activity – col. 9, li. 45-to col 10, li. 17; col. 11, lines 29-35; Fig. 4 – along with generation-contained callback constructs – see Fig. 4; *handle table*, *table_ptrs*, Fig. 6 col. 17 line 52 to col. 18, line 18 -- are replaced **reads on** identifying references of first code component); and

replacing the identified references to said first source code with references to said new source code component (e.g. col. 5 lines 50-52; next generation 110, Fig. 3; UPDATES: Add to next generation 90 – Fig. 3; Fig. 4; col. 11 line 29-67);

deallocating the first code component (*release ... rc_cleanup* - col 14 line 54 to col. 15 line 31).

Slingwine does not explicitly disclose replacing while said operating system remains active and while said operating system provides continual availability to applications of the hardware resources by applications operational in the computer system. However, Slingwine mentions about transparency to user in regard preventive and corrective actions, all of which being prohibitive because overhead or hardware implication (col. 4, lines 30-50), thus recommend a callback based approach to avoid deadlocks with minimum overhead resources (e.g. *no data locks, elimination of overheads ... repairing deadlocks* - col. 5, lines 57-63; see Fig. 2; *kernel running* - col. 10, li. 54-67), hence has suggested maintaining availability of resources

to the user applications while effectuating replacement at a low level being inexpensive in resources usage at a overall level. It would have been obvious for one skill in the art at the time the invention was made to implement the low overhead call back-based approach by Slingwine so that availability to user level applications of resources would not be impeded by the low overhead and dynamic replacement as purported above, because according to Slingwine, data can be maintained coherently at minimum cost to user level and the need to repair memory conflicts via undue resources expenses (as in deadlocks) as set forth above could be avoided.

As per claim 2, Slingwine discloses low-level interrupt (e.g. Fig. 5A, 5B, 5C, 5D; *interrupt* -- col. 18, lines 42-55 – Note: interrupt periodically invoked without need for user input suggests mutual exclusion of threads via callback implementation being low and transparent to application layer – see *kernel 36, interrupt* under *user process* – Fig. 2) but does not explicitly disclose method being transparent, however this transparency has been addressed as obvious from claim 1.

As per claim 3, Slingwine discloses wherein the method is scalable (e.g. *global generation, possible large number of processes* – see col. 18, lines 42-46; *expanded* – col. 17, lines 41-67; *per-thread where possible ... some other entity where possible* – col. 10, li. 44-47).

As per claim 4, Slingwine discloses a multiprocessor system with plurality of processors so that the method is implemented on each processor independently (e.g. *per-processor context* - Fig. 4; col. 19, li. 1-3).

As per claim 5, Slingwine discloses

transferring said identified references at said established quiescent state (e.g. *unlinking ... adds a callback to the next generation... links the new element ... erases the original element*

– col. 8 lines 54-67; col. 11 lines 10-17) from the first code component to the new code component; and

after transferring said identified references to the new code segment at said established quiescent state (e.g. *through a quiescent state ... allowing updater ... safely* - col. 10, lines 7-51; col. 8, lines 54-67), swapping the first code component with the new code component (e.g. col. 10, lines 7-51; col. 11 line 29-67 -- Note: context switching using callback transfer and quiescent state monitoring for switching from an old generation to a next generation by placing all list of callbacks into the next generation **reads on** swapping – see col. 11, lines 29-35; Fig. 4).

As per claim 6, Slingwine discloses separating the first code component into objects; and grouping said objects into a table (e.g. Fig. 6; *one bit per thread ... next level ... group of threads and so on* – col. 9, lines 31-44; col. 12, lines 5-15 – Note: thread and associated data structures per threat activity reads on objects and references thereto being grouped into structure – see col. 12, lines 35-53 – or pointer table – see Fig. 6-- to support *Updaters* - col. 17 line 52 to col. 18, line 18), arranging said grouped references to identify said references to said objects, which references to said objects are entered in the structure or call back table (e.g. col. 12, lines 5-15, 35-53; *handle table*, *table_ptrs*, Fig. 6; col. 17 line 52 to col. 18, line 18).

As per claim 7, Slingwine discloses wherein establishing includes:

establishing a quiescent state for the first code component, without locking the first code component, by tracking active threads to the first code component and identifying active threads as said references (refer to claim 1; col. 9 lines 34-41); and replacing includes:

transferring said identified references during the quiescent state from the first code component to the new code component; and

after transferring said identified references, swapping the first code component with the new code component (refer to claim 5).

As per claim 8, Slingwine discloses wherein the replacing step includes the steps of: establishing a quiescent state for the first code component that includes the identified references (refer to claim 7); transferring the identified references at the quiescent state from the first code component to the new code component by providing a infrastructure operating a best transfer algorithm (see Fig. 3-5 – Note: mutual exclusion policy via passing of changes as context structure updates to the next thread context reads on best algorithm infrastructure – refer to claim 6); and after transferring said quiescent state, swapping the first code component with the new code component (refer to claim 7); and further discloses transferring by providing an infrastructure to negotiate a best transfer algorithm (refer to claim 7 – Note: monitoring for a quescient state to provide data update to thread structures reads on negotiating for a best algorithm).

As per claim 9, Slingwine discloses wherein the step of identifying references includes the steps of

separating the first code component into objects, and grouping said objects into a table, arranging said table to identify said references to said objects, which identified references are entered in the table (refer to claim 6); and

the replacing step includes the steps of

establishing a quiescent state for the first code component that includes the identified references; transferring the identified references at the quiescent state from the first code component to the new code component by providing an infrastructure to negotiate a best transfer

algorithm; and after transferring said references to the new code, swapping the first code component with the new code component (refer to claims 5, 7, or 8).

As per claim 10, Slingwine discloses a system for swapping source code in a computer system including an operating system, said operating system including at least one source code component the system comprising means for:

instantiating an instance of a new code component (refer to claim 1) to replace said first code component;

establishing a quiescent state for the first code component (refer to claim 1);

transferring state from the first code component to the new code component (refer to claim 1); and

swapping the new code component for the first code component, including means for:

identifying, while said operating system is active and providing continual access to said resources, references to a first source code component of the operating system (re claim 1);

transferring the identified references to the new code component; and executing the replacing (re claim 5).

Slingwine does not explicitly disclose swapping while the OS providing continual availability to applications of hardware resources by applications operational in the computer system, and replacing while said operating system is active and providing continual access to said resources to said first source code with references to a new source code component for the operating system. However, the continual providing of resources to the user application while replacing is happening at a lower level has been rendered obvious in claim 1.

As per claim 11, refer to the rejection of claim 2-3.

As per claims 12-17, refer to claims 4-9, respectively.

As per claim 18, Slingwine discloses a program storage device, for use with a computer system including an operating system to provide access to hardware resources, wherein said operating system provides access to said resources via a first source code component (Fig. 1-2), said program storage device being readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for replacing said first source code component with a new source code component, the method steps comprising:

instantiating an instance of a new code component (refer to claim 1) to replace said first code component;

establishing a quiescent state for the first code component (refer to claim 1);

transferring state from the first code component to the new code component (refer to claim 1);

swapping the new code component including:

identifying references to said first source code component; and

replacing the identified references to said first source code with references to said new source code component;

deallocating the first code component.

all of which being addressed in claim 1.

Slingwine does not explicitly disclose replacing while said OS remains active in providing continual availability to applications of hardware resources by applications operational in the computer system; but this limitation has been addressed in claim 1.

As per claims 19-24, refer to claims 4-9, respectively.

Response to Arguments

14. Applicant's arguments filed 7/01/08 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 101 Rejection:

(A) Applicants have submitted that computer system in the preamble of claim 10 define hardware (Appl. Rmrks pg. 11, top). It is clear that the system as recited is for swapping code in a computer system, but in fact, this system comprises means for actions which are deemed mere software functionalities. The intended use is understood as for swapping code in a context of computer system with OS and code component; i.e. a computer context related to an intended use cannot be same as the very elements that comprise the system of the onset. The system is taken as a whole, and that whole includes a intended use and software elements. The rejection is maintained because the claimed system lacks hardware support to realize such functionality; and listing of software per se does not constitute one statutory category of subject matter.

USC § 112 Rejection:

(B) Applicants have submitted that paragraph 13, pg. 3 of the Specifications describe source code (Appl. Rmrks pg. 11, bottom) and this overcomes the § 112 Rejection. The rejection has pointed to specific parts of the hot swapping of an running OS where state or references are transferred to table. No *source code* per se is deemed part of the class/objects or instances of objects as these objects are identified in the transferring process. The source code at best appears to be a misuse of terminology, as objected to in the Specifications Objections. The claimed 'source code' further constitutes a mis-represented feature in view of the entire Disclosure or

lack of enablement; largely because how any form of *source code* is created and then turn executable for use in hot swapping is deemed not disclosed in sufficient implementation details.

USC § 103 Rejection:

(C) Applicants have submitted that mutual-exclusion in Swingline for concurrently updating current generation of threads with a next generation represents a important difference with the hot swapping of code component in O.S. by the invention. The step actions in the body of claims 1, 10, 18 amount how the setting of the preamble of these claims would be carried out. As amended, these claims introduce subject matter for which a new form of rejection would have to be necessitated; that is, the arguments regarded the amended claims would be deemed moot in view of the adjusted grounds of rejection. As for claims 1, 10, 18, each step action is deemed met by the corresponding actions in the method of Swingline; and whether the actions in Swingline relate to the mechanism of mutual exclusion, the language of the claimed limitations do not appear to categorically preclude mutual exclusion by any stretch of imagination; i.e. where exactly the steps as recited really prohibit state transfer or context switch of threads in a OS execution regarding thread contention (i.e. mutual exclusion) and swapping as by Swingline? Applicants' argument is deemed not sufficient to overcome the grounds of rejection, which is not a 102 Rejection, but rather a rationale of obviousness.

(D) The Applicants have submitted that (Appl. Rmrks pg. 15) as amended claims 1, 10, 18 depict quiescent state swapping to maintain OS continual provisioning (of HW resources) whereby well-performing code for a divergent set of applications is maintained. The argument is deemed a pleading that is not based on teachings interpreted from the claimed subject matter; i.e. it is clearly non-commensurate with the very language therein or rather an extra-curriculum type

of observation. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference.

The claims stand rejected as set forth in the Rejection.

Conclusion

15. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before

using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

September 08, 2008